Cross-site-scripting

Cross-site-scripting, or XSS as it is sometimes abbreviated to, is an attack that let's the attacker execute javascript code in the browser of the victim.

Finding XSS - Polyglot

```
jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert() )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=a]
```

https://github.com/0xsobky/HackVault/wiki/Unleashing-an-Ultimate-XSS-Polyglot

So, what's the worst that can happen?

The attacker is probably not that interested in changing the color or font of the website the victim is visiting. Although s/he could do that. The worst that can happen is probably the following:

1. Complete control over the browser The attacker can access plugins. Like password managers. The attacker can trick the user into allowing webcam or audio.
2. Session-hijacking/Cookie theft This is when the attacker steals the cookie that is saved in the browser. Using this cookie the attacker can log in to the service as the victim, and thereby gain access to his/her account. If the victim is an admin that has extended privileges (uploading code, images, or whatever) this could lead to a compromise of the server itself.
3. Keylogger The attacker can execute a keylogging-script that steals everything the user inputs in the website. This could be used to steal sensitive information, like passwords, credit cards information, chatlogs or whatever the user inputs.
4. Phishing The attacker can insert a fake login. Image that you visit a site, and from that site you are able to login using your facebook or google-account. The attacker could spoof that so that when you enter your credentials, they are then sent to the attacker.

Types of XSS

1. Persistent This is when the malicious code originates from the websites database. That means the attacker has managed to insert malicious code into the database. So every time the database serve that data the script will me executed. this is probably the most dangerous XSS, since it does not need to rely on social engineering.
2. Reflected This is an attack where the script originates from the users request. This might seem a bit illogical, why would a user inject malicious code to himself? Well the code can
3. DOM based DOM-based attacks are when something is injected into javascript on the DOM. So, it does not go by the server. Because the code gets executed in the response. Take a search-functionality for example. The users enters a search-parameter that gets sent to the server which might sanitize it or something. In the response the found search-items are sent, but not the search-query. But on the webpage the search query is exposed. "You searched for X" is shown. That is because it gets the search parameter from the url-parameter. By using `document.location.href` for example.

Protect yourself

The problem with XSS is that it is a bit hard for the users to protect themselves. If there is a problem witht the website there is not that much the user can do.

One can always use noscript to block all javascript code. But that pretty much destroys the whole experience with using the internet.

Protect your website

There are mainly two ways to protect against ** encoding ** and ** sanitizing **.

References:

http://brutelogic.com.br/blog/probing-to-find-xss/ http://excess-xss.com/